CS4677 Computer Forensics Forensics Duplication

Chris Eagle

Spring '06

References

- Text chapter 6-8
- Disk Imaging Specification 3.1.6
  - http://www.cftt.nist.gov/DI-spec-3-1-6.doc
- Computer Forensics Tool Testing Project
  - http://www.cftt.nist.gov/

Evidence Collection

- Duplicates are admissible as evidence
- Whether a system is off when we arrive or we shut it down after collecting any volatile evidence we need to properly image its hard drive

Types of Duplicates

- Forensic Duplicate
  - A file containing every bit of the source
  - No extra data
  - dd produces forensics duplicates
- Qualified Duplicate
  - May contain additional embedded information such as hash values
  - May compress empty sectors
  - SafeBack and EnCase produce these

Types of Duplicates (II)

- Restored Image
  - Restoration of a forensic or qualified image to another hard drive
    - Example: trying to create a bootable hard drive from a dd file
  - Tougher than it sounds
    - Works best if new hard drive is identical to original hard drive

Mirror Image

- Basically clones the original hard drive onto a new hard drive
  - Generally performed using a hardware drive duplicator
  - For best results new hard drive should be identical to original
  - Must make sure every bit is actually copied
    - There is a difference between cheap cloning hardware and forensically sound cloning hardware

Personal Preference

- Use dd/dcfldd to make an image of the hard drive
- Save the dd output as a file on a clean hard drive sufficiently large to hold the image
- No need to worry about matching original hardware
- Easier to make additional copies

Imaging Options

- Remove evidence drive from victim system
  - Make drive read only
    - Set read only jumper on drive if it has one
    - Use hardware "write blocker"
  - Install drive in imaging system
    - Preferably Linux as Windows will automount the drive
  - Image drive using dd onto media with sufficient free space

Imaging Options (II)

- Remove evidence drive from victim system
  - Use a hardware drive duplicator to mirror the drive
    - Use hardware that computes md5 or better yet SHA1 or SHA256 sums for you
    - WARNING – cheap drive duplicators are built for speed, not for forensics purposes and often attempt to copy only allocated portions of a disk.
  - Attempt to mirror onto identical media

Imaging Options (III)

- Use a bootable CD to boot victim system
  - Less desirable because mistakes could alter hard drive
  - Use dd on bootable CD to duplicate evidence drive to removable media (USB drive) or across a network (netcat)

Imaging Options (IV)

- Image while system is running
  - Last resort
  - Perform only if system is running when you arrive and you can’t shut it down
  - Use statically linked dd to image to removable media or across a network
  - Partition images may appear to have been "uncleanly mounted" when you go to analyze them

Evidence Integrity

- In all cases obtain both MD5 and SHA1, or better yet SHA256 hashes of your evidence as soon as possible
- Always best to compute at the same time the copy is being made
  - dcfldd can do this
  - Use `tee` to do this when using dd
  - Most hardware oriented forensics duplicators will generate at least one of these
  - Encase generates inline hashes

Wiping A Hard Drive

- Should you wish to delete old data (like an old case) from a hard drive
  - Install the drive in a Linux system
  - Use dd to overwrite the old data

    ```
    dd if=/dev/zero of=/dev/hdb
    ```

  - Some people talk of multiple overwrites or random data followed by all ones or all zeros
    - This is more for non-recoverability

dd Review

- Copies standard input to standard output by default
- Many command line arguments
- Three predominant versions
  - GNU dd
  - dcfldd
    - Defense Computer Forensics Lab dd
  - George Garner dd.exe for Windows

dd Summary

	O/S	Image Ram	Image Partitions	Image Drives	Compute Hash
dd	*nix, cygwin	Yes /dev/mem	*nix only	*nix only	No
dcfldd	*nix, cygwin	Yes /dev/mem	*nix only	*nix only	Yes
dd.exe	Windows only	Yes	Yes	Yes	MD5 only

dd Options

- if – input file
  - Can specify a file or a device name
  - Memory device
    - dd, dcfldd

      ```
      if=/dev/mem
      ```

    - dd.exe

      ```
      if=\\.\PhysicalMemory
      ```

dd Options (if)

- if – input file (cont)
  - Partition
    - A partition is a contiguous groups of sectors upon which a file system is created
    - *nix

      ```
      if=/dev/hdaX, if=/dev/hdbX
      if=/dev/sdaX
      ```

    - Windows (dd.exe only)

      ```
      if=\\.\C:
      if=\\?\Volume{690d4e02-00ae-11d8-aab0-806d6172696f}
         ^^^ "Volume Name:" from volume_dump
      ```

      - » Long name allows imaging of non-Windows partitions

dd Options (if)

- if – input file (cont)
  - Drive
    - A drive may contain many partitions as well as sectors that reside outside of any partition such as a boot sector and the partition table
    - *nix

      ```
      if=/dev/hda, if=/dev/hdb, …
      if=/dev/sda
      ```

    - Windows (dd.exe only)

      ```
      if=\\.\PhysicalDriveX
      ```

dd Options (of)

- of – output file
  - Usually specifies the destination file name
  - If omitted, output sent to stdout (which can be piped somewhere)
  - Will be same size as input file/device so make sure you have room

dd Options (bs)

- bs – block size
  - Basic size of an input chunk
  - Unit size to which the *count* argument applies
  - Default varies
    - dd, dcfldd – 512 bytes
    - dd.exe – 4096 bytes
  - For disk files/devices most efficient if bs == sector size or file system block size

dd Options (count)

- count – number of blocks to copy
  - NOT the number of bytes to copy (unless bs=1)
  - Total number of bytes will be lesser of
    - Input size
    - count * bs
  - Last block is not necessarily complete

dd Options (skip)

- skip – number of blocks to skip before beginning the copy operation
  - Useful when you do not wish to start at the beginning of a file/device
  - Examples
    - Pulling a specific file out of the middle of a disk image
    - Pulling a specific partition out of the middle of an image file

dd Options (seek)

- seek – the number of blocks to skip on the output device before writing begins
- Not generally used when creating forensics image files

dd Options (conv)

- conv – type of conversion to perform on input
  - Careful here, for forensics purposes, you generally do not want to alter the input data in any way
  - Could do ASCII to EBCDIC or upper to lower case among others

noerror

- Keep reading even if an error occurs
- Drops the current input block and proceeds to the next

sync

- Pad the failed input block with zeros
- This preserves size

```
conv=noerror,sync
```

dd Options (hashing)

- dcfldd

  `hash=NAME (either md5, sha1, sha256, sha384 or sha512)`

  `hashwindow=X`

  - Generates a hash after each X bytes as well as an overall hash
  - Set X to 0 to get only the overall hash
- dd.exe – MD5 only, across entire input set

  `--md5sum`

CS4677 Computer Forensics Evidence Handling

Chris Eagle

Spring '06

Evidence

- Anything that helps prove or disprove a point
  - Documents
  - Electronic media or files
  - Printouts
  - etc…

Original Evidence

- Original media associated with a computer/crime under investigation
- The first copy of perishable data
  - Volatile data from a live system
  - The output from network monitoring software

Best Evidence

- It is not always possible to confiscate all original evidence
- Federal Rules of Evidence (FRE) allow admission of duplicates
  - Rule 1001(3)
    - …An "original" of a photograph includes the negative or any print therefrom. If data are stored in a computer or similar device, any printout or other output readable by sight, shown to reflect the data accurately, is an "original".
  - Rule 1003. Admissibility of Duplicates
    - A duplicate is admissible to the same extent as an original unless (1) a genuine question is raised as to the authenticity of the original or (2) in the circumstances it would be unfair to admit the duplicate in lieu of the original.

Best Evidence (ii)

- Either
  - The original data if available
  - The original duplicate
- In either case chain of custody begins here
- A working duplicate of a piece of best evidence is NOT subject to chain of custody
  - It may be subject to validation

Working Copies

- Never perform examinations on best evidence
- Always create working copies
- Easiest if this can be done as the evidence is collected
  - Turn best evidence in to custodian
  - Keep working copy for examination
- Copies of copies are fine (and easier) as long as hashes match

Authentication

- The original collector of the evidence testifies
  - How the item was collected
  - That chain of custody was followed in its collection
- People that collect evidence may be called to testify, make sure they are competent

Validation

- This is why we compute hashes
- Must verify that the copies from which you derived your conclusions are identical to the best evidence
  - Obtain hashes on the best evidence at the earliest opportunity
  - Time stamp your hashes

Chain Of Custody

- Basically a paper trail documenting positive control of a piece of best evidence from the time of collection to its introduction in court
  - Designed to
    - Prevent access by unauthorized personnel
    - Prevent tampering

Evidence Handling Process

- Covered in the book Ch 6
- Summary
  - Photograph the scene
  - Document everything
    - Arrangement of components
    - Component manufacturer, model#, serial#
  - Label media as it is collected (case/item#)
  - Create an evidence tag for each individual item
  - Create backups and working copies of digital media
  - Transfer best evidence to evidence custodian

Evidence Tags

- Example in the book pg 167-169
- Contain
  - Who/how/when collected
  - Case number, item number
  - Description
  - Room for chain of custody
    - Details exactly who has handled the evidence and when

Evidence Custodian

- The person/people responsible for the storage of evidence
  - Should be the only people with access to the evidence storage area
    - Secure room or safe
- Must properly document all access to best evidence
  - Evidence log
- Responsible for periodic inventories of all evidence

CS4677 Computer Forensics File Systems

Chris Eagle

Spring '06

Reference

- Hard Drive basics
  - http://www.pcguide.com/ref/hdd/index.htm

Data Hierarchy

- Similar to the OSI network stack, data on a hard drive is layered to provide different levels of abstraction
  - Physical sectors
  - Partitions
  - Allocation units (blocks)
  - Space management layer (layout)
  - File layer (data)
  - Application layer (meaning)

Physical Layer

- Raw sectors on a disk
  - Created by the "low-level" formatting process
  - Generally 512 bytes
- Two addressing schemes
  - Cylinder/Head/Sector (CHS)
    - Must specify 3 parameters
    - Requires some knowledge of drive geometry
  - Logical Block Addressing (LBA)
    - Specify one number (0..MAX_SECTOR)
    - Drive geometry hidden by BIOS

Partitions

- Book calls this "data classification layer"
- Group consecutive sectors into units called partitions
  - Partition table keeps track of where and how large
  - Partition table resides at the end of the boot sector
  - Each partition has a partition-type ID
    - http://www.win.tue.nl/~aeb/partitions/partition_types-1.html
    - fdisk's l command

Partition Types

```
Command (m for help): l

 0  Empty           1c  Hidden Win95 FA 70  DiskSecure Mult bb  Boot Wizard hid
 1  FAT12           1e  Hidden Win95 FA 75  PC/IX           be  Solaris boot
 2  XENIX root      24  NEC DOS         80  Old Minix       c1  DRDOS/sec (FAT-
 3  XENIX usr       39  Plan 9          81  Minix / old Lin c4  DRDOS/sec (FAT-
 4  FAT16 <32M      3c  PartitionMagic  82  Linux swap      c6  DRDOS/sec (FAT-
 5  Extended        40  Venix 80286     83  Linux           c7  Syrinx
 6  FAT16           41  PPC PReP Boot   84  OS/2 hidden C:  da  Non-FS data
 7  HPFS/NTFS       42  SFS             85  Linux extended  db  CP/M / CTOS / .
 8  AIX             4d  QNX4.x          86  NTFS volume set de  Dell Utility
 9  AIX bootable    4e  QNX4.x 2nd part 87  NTFS volume set df  BootIt
 a  OS/2 Boot Manag 4f  QNX4.x 3rd part 8e  Linux LVM       e1  DOS access
 b  Win95 FAT32     50  OnTrack DM      93  Amoeba          e3  DOS R/O
 c  Win95 FAT32 (LB 51  OnTrack DM6 Aux 94  Amoeba BBT      e4  SpeedStor
 e  Win95 FAT16 (LB 52  CP/M            9f  BSD/OS          eb  BeOS fs
 f  Win95 Ext'd (LB 53  OnTrack DM6 Aux a0  IBM Thinkpad hi ee  EFI GPT
10  OPUS            54  OnTrackDM6      a5  FreeBSD         ef  EFI (FAT-12/16/
11  Hidden FAT12    55  EZ-Drive        a6  OpenBSD         f0  Linux/PA-RISC b
12  Compaq diagnost 56  Golden Bow      a7  NeXTSTEP        f1  SpeedStor
14  Hidden FAT16 <3 5c  Priam Edisk     a8  Darwin UFS      f4  SpeedStor
16  Hidden FAT16    61  SpeedStor       a9  NetBSD          f2  DOS secondary
17  Hidden HPFS/NTF 63  GNU HURD or Sys ab  Darwin boot     fd  Linux raid auto
18  AST SmartSleep  64  Novell Netware  b7  BSDI fs         fe  LANstep
1b  Hidden Win95 FA 65  Novell Netware  b8  BSDI swap       ff  BBT
```

Partition Table

```
Command (m for help): p

Disk /dev/hda: 80.0 GB, 80026361856 bytes
255 heads, 63 sectors/track, 9729 cylinders, total 156301488 sectors
Units = sectors of 1 * 512 = 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hda1   *        63    208844    104391   83  Linux
/dev/hda2        208845 101868164  50829660   83  Linux
/dev/hda3     101868165 105948674   2040255   82  Linux swap
/dev/hda4     105948675 156296384  25173855    f  Win95 Ext'd (LBA)
/dev/hda5     105948738 153163709  23607486   83  Linux
/dev/hda6     153163773 154207934    522081   83  Linux
/dev/hda7     154207998 156296384   1044193+  83  Linux
```

Allocation Layer

- This is where the O/S starts getting involved
- An allocation unit is the minimum unit the O/S can allocate
  - Windows – clusters
  - Unix – blocks
- One or more sectors
- Depends on partition size

Allocation Layer (ii)

- O/S evaluates trade offs
  - Large number of small units
    - Higher overhead
    - Less wasted space (slack space)
    - More disk i/o operations required
  - Smaller number of large units
    - Less overhead
    - More wasted space (slack)
    - Fewer i/o operations

Storage Space Management

- Free Space Tracking
- Allocation units are either free or in-use
- Status tracked differently by different O/S
  - DOS/Win95
    - File Allocation Table (FAT)
  - NTFS
    - Master File Table (MFT)
  - Unix
    - Superblock

File System Layer

- O/S provides two services
  - Files
    - Contain application layer data
  - Directories
    - Assignment of names to files
      - Maps a file's name to its location in a partition
    - Logical grouping of files

File System Structure (cont)

- File Indexing
  - Unix: Index Nodes (inode)
    - File names are held in directories which do nothing more than map a name to an inode
  - Windows
    - MFT
      - Small files (~1500 bytes) held entirely within the MFT
      - Large files use btree style allocation

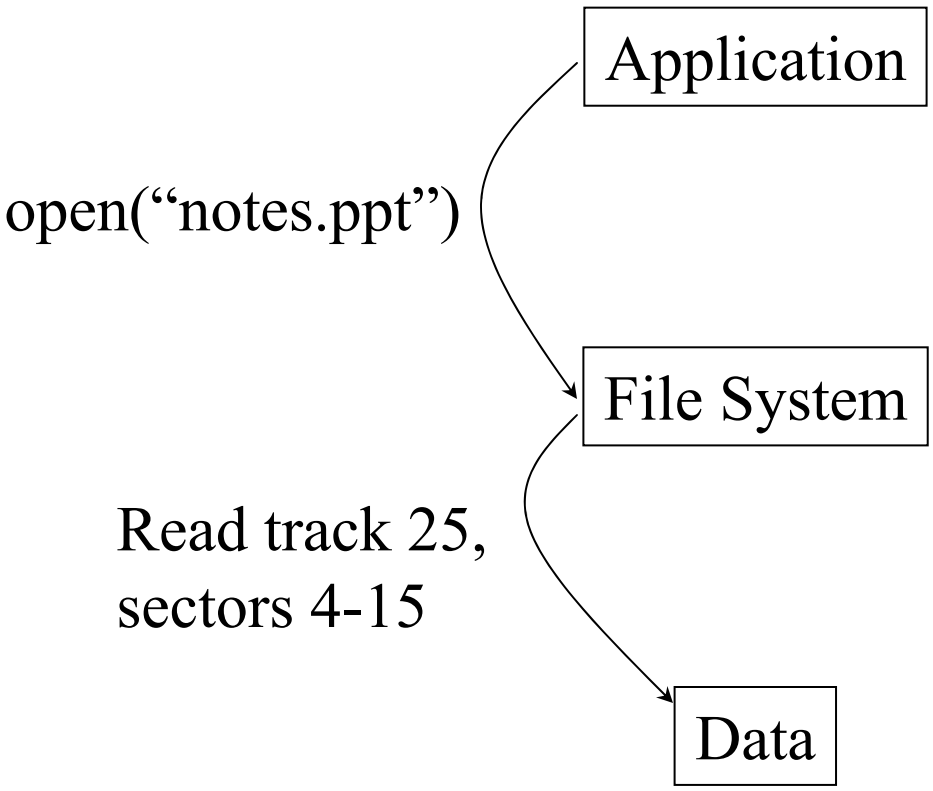
Application Layer

- The O/S could care less what is in a file
- Applications assign meaning to file content
- This has nothing to do with the name of the file
  - A file name is simply a hint as to its contents

File System Abstraction

- Data Layer
  - Lowest layer, sectors on a disk
  - Most disks are block devices so we get a minimum of a sector at a time
- File System Layer
  - Meta data used to index sectors into groups we call files
  - Translates from application layer “file view” to data layer track/sector view
- Application Layer
  - Each application has its own expectations for individual files
  - To be useful to an application, a file’s content must conform to the application’s format requirements

File System Hierarchy



File Creation

- Unix
  - A free inode is obtained from the superblock
  - File attributes are filled into the inode
  - Free blocks are requested from the superblock
  - Each block is recorded in the inode in the order in which it is obtained/used
    - Blocks need not be contiguous
- Windows is similar with interaction through the MFT

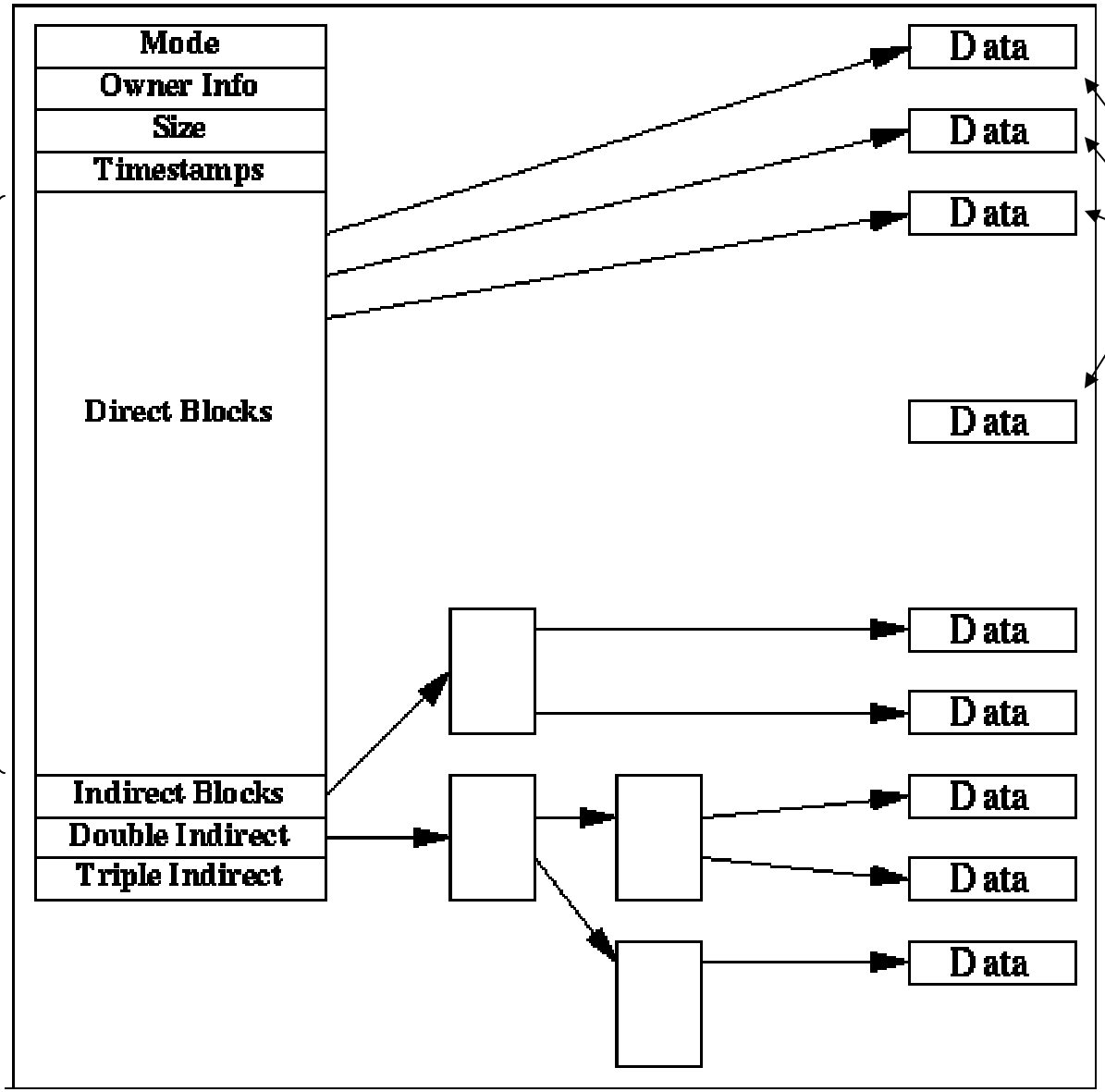
File Attributes

- Permissions
- MAC times
- Number of links
- Size
- Owner
- All stored in the file's index structure
  - Unix: inode
  - Windows: MFT

Unix inodes

- Inodes are a fixed size
  - Contain attribute data mentioned previously
  - Contain 15 pointers
    - 12 “direct” pointers to data blocks
    - 1 “single indirect” pointer
      - A pointer to a block of BLOCK_SIZE/4 direct pointers
    - 1 “double indirect” pointer
      - A pointer to a block of BLOCK_SIZE/4 single indirect pointers
    - 1 “triple indirect” pointer
      - A pointer to a block of BLOCK_SIZE/4 double indirect pointers

Linux inode


Mode
Owner Info
Size
Timestamps
Direct Blocks
Indirect Blocks
Double Indirect
Triple Indirect
Data
Data
Data
Data
Data
Data
Data
Data
BLOCK_SIZE
data blocks
12 direct
block pointers

debugfs

- A low level tool for poking around a file system
- Operates in interactive mode or batch mode
- Can display inode contents
- Can list deleted inodes
- Can display disk block contents
- Can dump blocks to new files

debugfs output

```
# debugfs -R show_super_stats /dev/hda6 | grep Block
debugfs 1.18, 11-Nov-1999 for EXT2 FS 0.5b, 95/08/09
Block size = 4096, fragment size = 4096

# debugfs -R "stat <340384>" /dev/hda6
debugfs 1.18, 11-Nov-1999 for EXT2 FS 0.5b, 95/08/09
Inode: 340384   Type: regular    Mode:  0664   Flags: 0x0
   Version/Generation: -1602593972
User:     0   Group:     0   Size: 17312
File ACL: 0    Directory ACL: 0
Links: 1   Blockcount: 40
Fragment:  Address: 0    Number: 0    Size: 0
ctime: 0x3e53cb37 -- Wed Feb 19 10:21:43 2003
atime: 0x3e53e31e -- Wed Feb 19 12:03:42 2003
mtime: 0x3e53cb37 -- Wed Feb 19 10:21:43 2003
BLOCKS:
690519 690520 690521 690522 690523
TOTAL: 5
```

File size

# of 512 byte blocks

Disk blocks used for this file

Unix inodes (cont)

- What is `BLOCK_SIZE`?
  - The minimum amount of space allocated to a file

```
debugfs -R show_super_stats dev | grep Block
```

- Small file (`<= 12 * BLOCK_SIZE`) will require only the direct pointers contained within an inode
  - Easy to recover
- Larger files will use first the single indirect pointer and then the double followed by the triple
  - More difficult to recover as you increase the levels of indirection

More debugfs

- Understanding block allocation

```
# debugfs -R show_super_stats /dev/hda6 | grep Block
debugfs 1.18, 11-Nov-1999 for EXT2 FS 0.5b, 95/08/09
Block size = 4096, fragment size = 4096

# ls -il tct-1.11.tar.gz
 340387 -rw-r--r--  1 root  root 314429 Feb 19 10:26 tct-1.11.tar.gz

# debugfs -R "stat <340387>" /dev/hda6 | grep TOTAL
debugfs 1.18, 11-Nov-1999 for EXT2 FS 0.5b, 95/08/09
TOTAL: 78
```

- Why does the file use 78 blocks instead of
  - $\lceil 314429 / 4096 \rceil = 77$ blocks?
- One block is dedicated to the single indirect block

Block Layout

- In an ideal world files remain unfragmented
  - All blocks allocated contiguously
- Indirect blocks are mixed with data blocks
  - Makes grabbing consecutive blocks to recreate the file a bit difficult
  - Must skip indirect blocks when grabbing blocks to recreate file
  - Need to know the structure of the file if possible

Unfragmented Files

- In this case files are laid out as follows
  - Let N = BLOCK_SIZE / 4
  - 12 data blocks
  - 1 indirect pointer block w/ N pointers
  - N indirect data blocks
  - 1 double indirect pointer block w/ N pointers
    - 1 indirect pointer block w/ N pointers
    - N double indirect data blocks
    - Repeat (N – 1) more times
  - 1 triple indirect pointer block w/ N pointers
    - 1 double indirect pointer block w/ N pointers
      - 1 indirect pointer block w/ N pointers
      - N triple indirect data blocks
      - Repeat (N – 1) more times
    - Repeat (N – 1) more times

Indirect Pointer Block

```
# dd if=/dev/urandom bs=4096 count=77 of=medium_file
77+0 records in
77+0 records out
# ls -il medium_file
5112112 -rw-r--r--  1 root root 315392 Oct 25 21:27 medium_file
# debugfs -R "stat <5112112>" /dev/hda1
debugfs 1.38 (30-Jun-2005)
Inode: 5112112   Type: regular    Mode:  0644   Flags: 0x0   Generation: 3581729
933
User:     0   Group:     0   Size: 315392
File ACL: 0    Directory ACL: 0
Links: 1   Blockcount: 624
Fragment:  Address: 0    Number: 0    Size: 0
ctime: 0x435f0596 -- Tue Oct 25 21:27:02 2005
atime: 0x435f0596 -- Tue Oct 25 21:27:02 2005
mtime: 0x435f0596 -- Tue Oct 25 21:27:02 2005
BLOCKS:
(0):10240376, (1-5):10240401-10240405, (6-11):10243492-10243497, (IND):10243498,
 (12-76):10243499-10243563
TOTAL: 78
```

```
# dd if=/dev/hda1 bs=4096 skip=10243498 count=1 | od -A x -t d4
1+0 records in
1+0 records out
000000    10243499    10243500    10243501    10243502
000010    10243503    10243504    10243505    10243506
000020    10243507    10243508    10243509    10243510
000030    10243511    10243512    10243513    10243514
000040    10243515    10243516    10243517    10243518
000050    10243519    10243520    10243521    10243522
000060    10243523    10243524    10243525    10243526
000070    10243527    10243528    10243529    10243530
000080    10243531    10243532    10243533    10243534
000090    10243535    10243536    10243537    10243538
0000a0    10243539    10243540    10243541    10243542
0000b0    10243543    10243544    10243545    10243546
0000c0    10243547    10243548    10243549    10243550
0000d0    10243551    10243552    10243553    10243554
0000e0    10243555    10243556    10243557    10243558
0000f0    10243559    10243560    10243561    10243562
000100    10243563           0           0           0
000110           0           0           0           0
*
001000
```

File Deletion

- The link count held within the inode is decremented
- IF the link count becomes zero
  - Each block listed in the file's inode is returned to the superblock for inclusion in the free block list
  - The inode itself is returned to the superblock's list of free inodes
- Neither the inode nor the blocks are overwritten!

debugfs again

- Can be used to recover deleted files (may not work on ext3 file systems)
  - lsdel command lists deleted inodes
  - cat command list file content associated with particular inode
  - dump command allows you to dump content to a new file
  - dumping a deleted inode will recover the deleted file

File Recovery

- Browse the list of free inodes for ones that contained data
  - On Linux these may have a dtime
- Browse the inode's list of blocks to see if they remain free
  - No guarantee that they still contain original data
  - A block may have been freed several times

CS4677 Computer Forensics Mounting Forensics Images

Chris Eagle

Spring '06

Imaging

- Partition
  - Usually done by software only
  - Use dd to create forensics duplicate
  - Name the partition to read and the file to save the image to

    ```
    dd if=/dev/hdb1 of=hdb1.img conv=noerror,sync
    ```
- Drive
  - Done with either hardware or software
  - Use dd to create forensics duplicate
  - Name the drive to read and the file to save the image to

    ```
    dd if=/dev/hdb of=hdb.img conv=noerror,sync
    ```

Evidence Protection

- Mounting an evidence file will change the file regardless of whether the file is not writeable or the file is mounted read only.
- For image files, set the immutable bit on the file using chattr

```
chattr +i <image file>
```

  - Prevents inadvertent changes to image
  - May not be able to mount

Loopback Device

- Linux capability
- Allows binary images of a file system to be mounted just like a physical device/partition
- Perfect for mounting forensics images for analysis
- Name: `/dev/loopN`
- `mount` command can do all the work for us
  - Occasionally fails for no apparent reason
  - Use `losetup` for finer control

Mounting a Partition Image

- Images of partitions contain entire file systems (ext2, ntfs, fat, …)
- Conveniently, the `mount` command can only mount file systems
- Using mount to mount an image

```
mount -o ro,loop,noexec,noatime hdb1.img /mnt/evidence
```

  - `ro`: read only
  - `loop`: use next available loopback device
  - `noexec`: don't allow execution of any binaries in the file system
  - `noatime`: don't atimes of any files in the file system

Using losetup

- Sometimes mount chokes with the loop option
- Use losetup instead
  - `losetup /dev/loop0 hdb1.img`
    - Associated loopback device zero with the image file hdb1.img
  - `mount -o ro,noexec,noatime /dev/loop0 /mnt/evidence`
    - Mounts the "device" /dev/loop0
  - When complete
    - `umount /dev/loop0`
      - Unmount the device
    - `losetup -d /dev/loop0`
      - Detach /dev/loop0 from hdb1.img

Drive Images

- Can’t mount a drive, only partitions
- Three options
  - Strip out partitions using dd
    - Described here:
      - http://sleuthkit.sourceforge.net/informer/sleuthkit-informer-2.html#split
  - Use the *offset* parameter to losetup
    - Drawback is that you can't recognize the end of the partition
  - Use NASA loopback drivers
    - These allow you to mount a drive image
    - ftp://ftp.hq.nasa.gov/pub/ig/ccd/enhanced_loopback/
      - You are downloading an actual kernel, not just a driver

Extracting Partition Images

- Basic idea
  - Use fdisk on the image to read the partition table
  - With information from fdisk, use dd to extract the portions of the file corresponding to each partition

Using fdisk

```
[root@eaglepc cs4677]# fdisk -lu /dev/hda

Disk /dev/hda: 13.0 GB, 13022324736 bytes
255 heads, 63 sectors/track, 1583 cylinders, total 25434228 sectors
Units = sectors of 1 * 512 = 512 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *          63       48194       24066   83  Linux
/dev/hda2           48195    24900749    12426277+  83  Linux
/dev/hda3        24900750    25430894      265072+  82  Linux swap
```

- -u option causes display to be in units of sectors

fdisk on an Image File

```
[root@eaglepc cs4677]# fdisk -lu hda.img
You must set cylinders.
You can do this from the extra functions menu.

Disk hda.img: 0 MB, 0 bytes
255 heads, 63 sectors/track, 0 cylinders, total 0 sectors
Units = sectors of 1 * 512 = 512 bytes

  Device Boot    Start       End    Blocks   Id  System
hda.img1   *        63     48194     24066   83  Linux
hda.img2         48195  24900749  12426277+  83  Linux
Partition 2 has different physical/logical endings:
     phys=(1023, 254, 63) logical=(1549, 254, 63)
hda.img3      24900750  25430894    265072+  82  Linux swap
Partition 3 has different physical/logical beginnings (non-Linux?):
     phys=(1023, 254, 63) logical=(1550, 0, 1)
Partition 3 has different physical/logical endings:
     phys=(1023, 254, 63) logical=(1582, 254, 63)
```

- Because fdisk does not know the geometry

Educating fdisk

- Tell fdisk about the drive geometry

```
[root@eaglepc cs4677]# fdisk -lu -C 1583 -S 63 -H 255 hda.img

Disk hda.img: 0 MB, 0 bytes
255 heads, 63 sectors/track, 1583 cylinders, total 0 sectors
Units = sectors of 1 * 512 = 512 bytes

   Device Boot      Start         End      Blocks   Id  System
hda.img1   *          63       48194       24066   83  Linux
hda.img2           48195    24900749    12426277+  83  Linux
hda.img3        24900750    25430894      265072+  82  Linux swap
```

Extracting a Partition

- With the information from fdisk, you can use dd to extract a portion of a file
  - skip into the file to start at the first sector of the partition
  - count only as many sectors as you need
  - bs set to sector size
- Example – grabbing /dev/hda1

```
dd if=hda.img of=hda1.img skip=63 count=48132 bs=512
```

Using losetup

- losetup can be told to offset into a file to find the start of your data
  - -o option specifies number of ***bytes*** to offset
- Previous example
  - Partition 1 begins at sector 63 = 32256 bytes
  - `losetup –o 32256 /dev/loop0 hda.img`
  - `mount –o ro,noexec /dev/loop0 /mnt/evidence`
- Problem is that end of partition is not recognized